Une critique de la notion de test de processus fondée sur la non séparabilite de certaines classes de langages

Philippe Darondeau

IRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tel.(3) 954 90 20

Rapports de Recherche

N° 425

UNE CRITIQUE DE LA NOTION DE TEST DE PROCESSUS FONDÉE SUR LA NON SÉPARABILITÉ DE CERTAINES CLASSES DE LANGAGES

Philippe DARONDEAU

Juillet 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

UNE CRITIQUE DE LA NOTION DE TEST DE PROCESSUS FONDEE SUR LA NON SEPARABILITE DE CERTAINES CLASSES DE LANGAGES

Philippe DARONDEAU
IRISA, Campus de Beaulieu
F35042 RENNES CEDEX

Publication Interne n° 259
Juin 1985
40 pages

RÉSUMÉ Une classe de langages est dite séparable s'il existe pour toute paire de langages distincts L' et L" un langage L tel que $L \cap L' = \emptyset \longleftrightarrow L \cap L" \neq \emptyset$. Nous résolvons par la négative la question de la séparabilité pour un large éventail de classes de langages infinitaires, de Alg^{ω} à $\sum_1^1$. Nous prouvons ensuite, pour le calcul des systèmes communicants, qu'il n'existe pas de notion de test qui sépare tous les agents différant par leurs suites infinies d'actions visibles.

ABSTRACT A class of languages is separable if, for each pair of non identical languages L' and L", there exists some language L such that $L \cap L' = \emptyset \longleftrightarrow L \cap L" \neq \emptyset$. We give a negative answer to the question of separability for a large variety of classes of infinitary languages, from Alg^{ω} to $\sum_1^1$. We then prove that there exists for the calculus of communicating systems no notion of testing which can separate every pair of agents which differ by their infinite sequences of visible actions.

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L.A. 227)
UNIVERSITÉ DE RENNES 1 I.N.S.A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

UNE CRITIQUE DE LA NOTION DE TEST DE PROCESSUS FONDEE SUR LA NON SEPARABILITE DE CERTAINES CLASSES DE LANGAGES

I. INTRODUCTION

Un processus peut être défini comme un ensemble de suites d'actions allié à un dispositif de production de ces suites, déterministe si l'ensemble est un singleton ou non déterministe. Dans la vision traditionnelle des processus, les actions sont présumées autonomes et le dispositif de production opère indépendamment de tout environnement. En extension, un processus se confond avec un langage, finitaire ou infinitaire, image de l'ensemble de ses suites d'actions par un morphisme qui efface les actions réputées invisibles, et laisse les autres actions inchangées (si l'on suppose une interprétation libre de ces dernières). Dans la vision des systèmes communicants introduits par R.Milner [11], les processus sont de nature plus complexe: le dispositif de production est asservi à un environnement, qu'il conditionne à son tour par le jeu des potentialités d'actions. En extension, un processus est un ensemble de comportements observables, chacun caractérisé par une suite d'actions visibles et par un ensemble de conditions sur l'environnement, nécessaires à la production de cette suite.

Le propos de la présente étude est de déterminer, dans chacun des deux cadres précédents, en quelle mesure des processus qui diffèrent par leur extension peuvent être distingués par des ensembles de tests de même nature que ces processus.

S'agissant de processus autonomes, nous suggérons $|p| \cap |t| \neq \emptyset$ comme critère de la satisfaction du test t par le processus p, où $|\ |$ indique l'extension. Le problème ci-dessus est donc un problème de séparabilité. Pour chaque classe de langages définie par un type particulier de dispositif de production, le problème s'énonce ainsi: étant donnés deux langages L' et L" distincts l'un de l'autre, est-il toujours possible de trouver un langage L tel que les assertions $(L \cap L' = \emptyset)$ et $\sim (L \cap L'' = \emptyset)$ soient équivalentes ? Une notion de séparabilité forte peut également être considérée: étant donnés un langage L et un mot u n'appartenant pas à L, peut-on toujours trouver un langage L_u contenant u et disjoint de L ? (Les classes de langages n'étant généralement pas fermées pour l'union dénombrable, ces questions de séparabilité faible ou forte ne sont pas de nature topologique). Nos problèmes de séparabilité ont des solutions immédiates pour les classes de langages finitaires, et nous ne les poserons dans la suite que pour des langages purement infinitaires. Dans ce cadre restreint, la séparation entre L' et L" resp. entre u et L est immédiate si $Adh(L') \neq Adh(L'')$ resp. si $u \notin Adh(L)$, où Adh(L) est la clôture de L dans l'espace métrique complet induit par la distance ultramétrique sur les mots [1]. Curieusement cependant, la quasi totalité des classes de langages infinitaires que nous étudierons sont non séparables. Ce résultat négatif vaut pour la plupart des classes construites par fermetures de Kleene [3]. Il vaut aussi pour la classe $\sum_1^1$ [12], ce qui laisse présager l'insuffisance de toute notion de test pour toute espèce de processus suffisamment généraux.

Dans le cadre des processus communicants, nous limiterons notre étude à la classe connue sous l'appellation C.C.S. Nous nous proposons d'étendre légèrement les notions de test et d'équivalence de test introduites par M. Hennessy et R. de Nicola [6]. Selon ces auteurs, un test est un processus qui s'exécute en parallèle avec le processus testé, les deux processus communiquant et formant un système isolé. Dans ce système où chaque processus est l'environnement de l'autre, certaines exécutions du processus de test sont réputées gagnantes. Le critère de succès est un critère finitaire. Le critère d'échec est infinitaire: une exécution du

processus de test échoue lorsque aucun de ses préfixes n'indique le succès. Deux processus sont équivalents si pour tout test t, chacune des assertions "x satisfait t dans toute exécution" et "x ne satisfait t dans aucune exécution" est simultanément vérifiée ou falsifiée par ces deux processus. Fondés sur cette équivalence de test $\underline{\sim}$ ont été proposés pour C.C.S. et pour sa variante synchrone S.C.C.S. des modèles dénotationnels ayant la propriété de pleine abstraction [10], i.e. tels que les significations $\mathcal{M}(p)$ et $\mathcal{M}(q)$ calculées pour p et q soient égales s s i $C[p] \underline{\sim} C[q]$ pour tout contexte C de la classe considérée (C.C.S. dans [6], S.C.C.S. dans [7]). La notion de test apparait ainsi dans sa fonction première qui est celle de critère de validité d'un modèle sémantique. C'est ce critère que nous nous proposons de discuter en montrant qu'un modèle sémantique pleinement abstrait vis à vis d'une équivalence de test identifie nécessairement certains processus qui diffèrent par leurs suites infinies d'actions visibles. Nous nous appuierons sur la non séparabilité de $\sum_1^1$ pour établir ce résultat, valable pour des tests généralisés et sous toutes les hypothèses plausibles d'équité ou de non équité. Le résultat annoncé indique une certaine forme d'incohérence pour toute équivalence de test reposant en partie sur la détection de propriétés des calculs ininitaires. L'équivalence d'observation à la base du modèle construit dans [2] esquive précisément cette critique.

La suite du texte est organisée en deux parties. Le chapitre 2 étudie la séparabilité des langages. Le chapitre 3 étudie la séparabilité de la classe des processus C.C.S. Une brève conclusion (§4) souligne, à la lumière des résultats, les limitations inhérentes aux modèles du parallélisme construits sur des domaines algébriques.

II. SEPARABILITE DE CLASSES DE LANGAGES

Après quelques rappels et définitions (§2.1), nous nous intéressons aux classes de langages obtenues par fermeture au sens de Kleene des classes de la hiérarchie de Chomsky (§2.2), puis à d'autres classes de langages faisant intervenir des opérateurs de mélange finitaire ou infinitaire (§2.3), enfin à la classe $\sum_1^1$ (§2.4).

II.1 - NOTATIONS ET DEFINITIONS

Pour tout alphabet dénombrable X, X^* est le monoïde libre sur X, d'élément neutre ε, et X^ω est l'ensemble des mots infinis sur X. On note $^\omega$ l'application de $(X^* - \{\varepsilon\})$ dans X^ω qui au mot fini u fait correspondre le mot infini u^ω limite de la suite $(u^i)_{i>0}$. Pour tout couple de mots (u,v) dans $X^* \times (X^* \cup X^\omega)$, uv est le mot obtenu par concaténation de u à gauche de v. Les opérations précédentes se généralisent aux ensembles de mots, ou langages. Une classe de langages sur X est finitaire si tout langage de la classe est un sous-ensemble de X^*, elle est (purement) infinitaire si tout langage de la classe est un sous-ensemble de X^ω. Après ces quelques rappels, nous introduisons deux notions de séparabilité applicables aux classes de langages infinitaires.

Définition: Soit $\mathcal{L}$ une classe de langages infinitaires sur un alphabet X. $\mathcal{L}$ est **fortement séparable** si pour tout L dans $\mathcal{L}$ et pour tout u dans $X^\omega - L$, il existe L_u dans $\mathcal{L}$ tel que $u \in L_u \subseteq X^\omega - L$. $\mathcal{L}$ est (faiblement) **séparable** si pour tout doubleton $\{L', L''\}$ dans $\mathcal{L}$, il existe L dans $\mathcal{L}$ tel que $L \cap L' = \emptyset \iff L \cap L'' \neq \emptyset$.

Nous résolvons ci-dessous la question de la séparabilité pour diverses classes de langages infinitaires, en partant des plus classiques.

II.2 - CLOTURES DE KLEENE DES FAL DE LA HIERARCHIE

Soit $\mathcal{L}$ une classe de langages finitaires sur un alphabet X. Nous notons $\mathcal{L}^\omega$ la fermeture de $\mathcal{L}$ au sens de Kleene, ou ensemble des langages définis par des expressions de la forme :

$$\bigcup_{i=1}^{n} A_i (B_i - \{\varepsilon\})^\omega \quad , \quad A_i \in \mathcal{L} \, , \, B_i \in \mathcal{L} \, .$$

$\mathcal{L}^\omega$ Nous posons dans cette section la question de la séparabilité de pour $\mathcal{L}$ variant dans la hiérarchie des familles agréables de langages (FAL). Rappelons qu'une FAL est une classe de langages finitaires fermée pour les opérations d'union ensembliste finie, de produit de concaténation et d'étoile (*) ainsi que pour les homomorphismes alphabétiques et leurs inverses. Parmi les FAL les plus connues, citons les familles Rat et Alg des langages rationnels resp. algébriques, la famille Etol des langages engendrés par tableaux de substitution parallèle, les familles Rec et Recenum des langages récursifs resp. récursivement énumérables. Selon la hiérarchie d'inclusion, ces FAL sont ordonnées comme suit :

$$\text{Rat} \subset \text{Alg} \subset \text{Etol} \subset \text{Rec} \subset \text{Recenum}$$

Les deux théorèmes suivants règlent les problèmes de séparabilité pour cette hiérarchie.

théorème Rat^ω est fortement séparable

preuve ... car fermée par complémentation d'après les théorèmes de Büchi et de Mac Naughton (cf. [3]) □

théorème Si $\mathcal{L}$ contient Alg, $\mathcal{L}^\omega$ n'est pas séparable.

preuve

Le principe est de construire un emboitement de langages algébriques infinitaires ayant même ensemble de mots ultimement périodiques. (On rappelle qu'un mot ultimement périodique est la limite d'une suite non stationnaire de mots finis $(uv^i)_{i>0}$). Etant données deux lettres a, b de l'alphabet, soient E et F les deux langages algébriques donnés par :

$$E = a^+b,\ F = \bigcup_{n>0} E^* a^n b\, E^* a^n b .$$

Clairement, E^ω et F^ω ont même ensemble de mots ultimement périodiques, et F^ω est inclus dans E^ω. Mais E^ω diffère de F^ω, car $aba^2ba^3b... \notin F^\omega$. Soit Fin l'ensemble des langages finitaires.

Supposons l'existence dans Fin^{ω} d'un langage L qui sépare E^{ω} de F^{ω}, c'est à dire vérifie $L \cap F^{\omega} = \emptyset$ et $L \cap E^{\omega} \neq \emptyset$. Nous montrons ci-dessous que cette supposition est absurde.

Soit $L = \bigcup_{i=1}^{m} A_i B_i^{\omega}$ où $A_i, B_i \in Fin$, et soit $w \in L \cap E^{\omega}$. Par définition de L, il existe $i \leq m$, $u \in A_i$ et une suite de mots $v_j \in B_i$, $j > 0$, tels que $w = uv_1v_2\ldots v_j\ldots$.

Puisque w appartient à E^{ω}, il existe certainement deux entiers k et l ($k < l$) tels que $v_{k+1}\ldots v_l$ contienne la lettre b et commence ou finisse par la lettre a : si un facteur $v_p\ldots v_q$ de w se termine par b, la lettre initiale de v_{q+1} est a.

Posons $f = uv_1\ldots v_k$ et $g = v_{k+1}\ldots v_l$.

Par construction, fg^{ω} est un mot ultimement périodique commun aux langages $A_iB_i^{\omega}$ et E^{ω}. Comme E^{ω} et F^{ω} ont même ensemble de mots ultimement périodiques, fg^{ω} appartient à l'intersection des langages L et F^{ω}, d'où la contradiction □

Deux voies restent ouvertes pour découvrir d'autres classes séparables de langages sans modifier radicalement leur procédé de formation: appliquer la fermeture de Kleene à des familles (finitaires) incomparables avec Alg, ou introduire une variante de cet opérateur de fermeture. Ces deux voies sont explorées dans la section suivante.

II.3 - OPERATEURS DE MELANGE

Nous étudions successivement la fermeture de Kleene de Conc, une classe de langages concurrents qui est construite à l'aide de deux opérateurs de mélange fini [13], puis une variante $Alg^{\textcircled{\omega}}$ de Alg^{ω} construite à partir de Alg à l'aide d'un opérateur de mélange infini $\textcircled{\omega}$.

Définition : Conc = $(\cdot, \cup, *, \odot, \circledast)$ $(X \cup \{\varepsilon\})$ est la classe des langages engendrés à partir du mot vide et des lettres de l'aphabet X par les opérateurs concurrents de mélange ($\odot$) et de mélange itéré ($\circledast$) joints aux opérateurs rationnels $(\cdot, \cup, *)$. Les opérateurs concurrents sont définis de la façon suivante sur $P(X^*)$:

$$A \odot B = \{u_1 v_1 \ldots u_n v_n \mid u_i, v_i \in X^*,\ u_1 \ldots u_n \in A,\ v_1 \ldots v_n \in B\},$$

$$A^{\circledast} = \bigcup_{i \geq 0} A^{\textcircled{i}},\quad A^{\textcircled{0}} = \{\varepsilon\},\quad A^{\textcircled{i+1}} = A^{\textcircled{i}} \odot A .$$

Propriétés : Conc est une FAL, Alg $\not\subset$ Conc, Conc $\not\subset$ Alg [5] .

Théorème : Si $\mathcal{L}$ contient Conc, $\mathcal{L}^{\omega}$ n'est pas séparable .

Preuve :

L'idée est ici encore de construire sur deux lettres a et b un emboitement de langages concurrents (infinitaires) ayant même ensemble de mots ultimement périodiques. La définition de ces langages fait intervenir les constantes suivantes, à valeur dans Conc :

$$A = ((ab)^{\circledast} \odot a^{+})$$
$$B = ((ba)^{\circledast} \odot b^{+})$$
$$C = (a \odot b)^{\circledast}$$
$$D = b(a \odot b)^{\circledast} a$$
$$E = a(a \odot b)^{\circledast} b$$

Les langages L et L' que nous allons considérer sont donnés par

$$L = (a^*b^*)^* (A^{\omega} \cup B^{\omega} \cup C^{\omega})$$

$$L' = (a^*b^*)^* (A^{\omega} \cup B^{\omega}) \cup ((a \odot b)^{\circledast} \odot (a^{+} \cup b^{+})) (D^{\omega} \cup E^{\omega}).$$

Clairement, L' est inclus dans L, car D et E sont deux sous-ensembles de C qui n'est autre que le Dyck généralisé sur deux lettres. Réciproquement, L n'est pas inclus dans L' : le mot $aba^2b^2\ldots$ est dans $C^\omega - L'$ (lemme 1).

Soit Fin l'ensemble des langages finitaires.

Supposons l'existence de F et G dans Fin vérifiant $FG^\omega \cap L \neq \emptyset$ et $FG^\omega \cap L' = \emptyset$. Soit alors $w \in (L-L')$ un mot admettant la décomposition $uv_1v_2\ldots v_i\ldots$, $u \in F$ et $v_i \in G$, $i > 0$. Nous poursuivons par analyse de cas.

*Cas 1 $(\exists\ i)\ (|v_i|_a > |v_i|_b)$

Soit v l'un des préfixes de v_i pour lequel la valeur $|v|_a - |v|_b$ est minimale, soit $v_i = vv'$ et soit $u' = uv_1\ldots v_{i-1}$.

Alors $v'v \in A$ et $u'v(v'v)^\omega \in L'$, donc $u'v_i{}^\omega \in FG^\omega \cap L'$, ce qui est contradictoire.

*Cas 2 $(\exists\ i)\ (|v_i|_b > |v_i|_a)$

Ce cas est semblable au précédent.

*Cas 3 $(\forall_i)\ (|v_i|_a = |v_i|_b)$

Soit v l'un quelconque des mots v_i non vides, alors $uv^\omega \in FG^\omega \cap L'$ (lemme 2), ce qui est à nouveau contradictoire □

lemme 1 : Soit $w = aba^2b^2\ldots$, alors $w \notin L'$.

preuve du lemme

Tout facteur u de w dans A est de la forme $u = a^i b^i \ldots a^j b^j a v$, où $v \in a^*$ et $0 \leqslant i \leqslant j$. La concaténation $u_1 \ldots u_n$ de n mots de cette forme ne peut résulter en un facteur de w que si tous les u_i sauf éventuellement u_1 sont des mots sans b. Or tout facteur droit de w contient une infinité d'occurences b. Par conséquent, $w \notin (a^*b^*)^* A^\omega$.

Tout facteur u de w dans B est de la forme $u = b^i a^j$, $0 < i$ et $0 \leqslant j \leqslant i-1$. La concaténation $u_1 \ldots u_n$ de n mots de cette forme ne peut résulter en un facteur de w que si tous les facteurs u_i sauf éventuellement u_n sont des mots sans a, or tout facteur droit de w contient une infinité d'occurences a. Par conséquent, $w \notin (a^*b^*)^* B^\omega$.

Supposons $w = uv$ et $v \in D^\omega$, alors $v = v_1 v_2 v_3 \ldots$ où les v_i sont des mots de D. Par définition de w, puisque v_1 finit par une occurence a et que v_3 commence par une occurence b, v_2 admet une décomposition de la forme $v_2 = b^i a^{i+1} \ldots b^{i+k} a^{i+k+1}$, $k \geqslant 0$. On a $|v_2|_a = |v_2|_b + k + 1 > |v_2|_b$ donc $v_2 \notin C$. Comme D est inclus dans C, $v_2 \notin D$ (contradiction).

Supposons $w = uv$ et $v \in E^\omega$, alors $v = v_1 v_2 v_3 \ldots$ où les v_i sont des mots de E. Par définition de w, puisque v_1 finit par une occurence b et que v_3 commence par une occurence a, v_2 admet une décomposition de la forme $v_2 = a^i b^i \ldots a^{i+k} b^{i+k}$. Le mot v_1, qui est à la fois facteur de w et élément du Dyck généralisé, présente alors une forme semblable. Par conséquent, u s'écrit $aba^2b^2 \ldots a^j b^j$, d'où

$$|u|_a = |u|_b \quad \text{et} \quad u \notin (a \odot b)^{\circledcirc} \odot (a^+ \cup b^+) \qquad \square$$

lemme 2 : Soit v un mot non vide sur les lettres a et b, tel que $|v|_a = |v|_b$, alors $uv^\omega \in L'$ pour tout $u \in \{a,b\}^*$.

preuve du lemme

Le mot v^2 admet par hypothèse une décomposition $v^2 = fa^x b^y a^z g$ pour laquelle x, y, z sont tous non nuls.

Si $|uf|_b \neq x + |uf|_a$, posons $u' = ufa^x$ et

$$v' = b^y a^z g f a^x$$

alors u' et v' appartiennent respectivement à $((a \odot b)^{\circledast} \odot (a^+ \cup b^+))$ et à D. Puisque $uv^\omega = u'v'^\omega$, $uv^\omega \in L'$.

Si $x + |uf|_a \neq y + |uf|_b$, posons $u' = ufa^x b^y$ et

$$v' = a^z g f a^x b^y$$

alors $uv^\omega = u'v'^\omega$ et $u'v'^\omega \in L'$ pour les mêmes raisons que ci-dessus.

Finalement, l'un des deux cas précédents est nécessairement vérifié, car la propriété $y \neq 0$ rend impossible la double égalité

$$|uf|_b = x + |uf|_a = y + |uf|_b \qquad \boxtimes$$

Définition : Pour tout alphabet X, nous notons $^{\circledcirc}$ l'opérateur de $P(X^+)$ dans $P(X^\omega)$ qui à tout langage finitaire L (dépourvu du mot vide) fait correspondre le langage infinitaire $L^{\circledcirc}$ déterminé par l'ensemble

$$\{w \in X^\omega \mid \exists f : \mathbb{N} \to (\mathbb{N} \to (X \cup \{\varepsilon\})) : f(i) = f_i,\ \exists g : \mathbb{N} \to \mathbb{N},$$

$$[w = \lim_j (f_{g(0)}(0) f_{g(1)}(1) \ldots f_{g(j)}(j))] \ \&$$

$$[\forall i, \forall j, f_i(j) \neq \varepsilon \Leftrightarrow g(j) = i] \ \&$$

$$[\forall i, \lim_j (f_i(0) f_i(1) \ldots f_i(j)) \in L]\} \ .$$

Définition : Pour tout alphabet X, $\odot$ est l'opérateur de $P(X^*) \times P(X^\omega)$ dans $P(X^\omega)$ qui au couple A, B fait correspondre le langage $A \odot B$ déterminé par l'ensemble :

$$\{w \in X^\omega \mid \exists f,g : \mathbb{N} \to (X \cup \{\varepsilon\}),\ \exists h : \mathbb{N} \to \{f,g\} : h(i) = h_i\ ,$$

$$[w = \lim_j (h_0(0)\ h_1(1)\ldots h_j(j))] \ \&$$

$$[\forall i,\ (h(i) = f \Leftrightarrow f(i) \neq \varepsilon)\ \&\ (h(i) = g \Leftrightarrow g(i) \neq \varepsilon)] \ \&$$

$$[\lim_j (f(0)\ldots f(j)) \in A \ \ \&\ \ \lim_j (g(0)\ldots g(j)) \in B]\}.$$

Définition : Pour toute classe $\mathcal{L}$ de langages finitaires on note $\mathcal{L}^\odot$ la classe des langages infinitaires décrits par des expressions de la forme

$$\bigcup_{i=1}^{n} A_i\ op_i^1\ (B_i\ op_i^2\ (C_i - \{\varepsilon\})^\odot)$$

où $\{op_i^1, op_i^2\} = \{\cdot, \odot\}$ et A_i, B_i, C_i varient dans $\mathcal{L}$ pour tout $i \in \{1\ldots n\}$

Théorème : Si $\mathcal{L}$ contient Alg, $\mathcal{L}^\odot$ n'est pas séparable .

Preuve

On considère les langages suivants sur {a, b, c} :

$$L = (ca^+b)^\odot \cup \{a, b, c\}^*\ ba\{a, b, c\}^\odot$$

$$L' = A \odot (ca^+b)^\odot \cup \{a, b, c\}^*\ ba\{a, b, c\}^\odot$$

où $A = \bigcup_{n>0} (ca^nb \odot ca^nb)$.

On montre aisément que A est algébrique (lemme 3); L' est par ailleurs inclus dans L (car $B^{@} = B^{n}_{:} B^{@}$ pour tout n), mais l'inclusion opposée est invalide puisque le mot $cabca^2bca^3b\ldots$ n'appartient pas à L'.

Supposons l'existence de trois langages finitaires F, G, H et d'un mot w dans $\{a,b,c\}^{\omega}$ tels qu'en posant

$$L'' = F\ op^1\ (G\ op^2 (H - \{\varepsilon\})^{@}),$$

où $\{op^1, op^2\} = \{\cdot, :\}$, on obtienne

$$L' \cap L'' = \emptyset \text{ et } w \in (L \cap L'').$$

Nécessairement, H contient des mots présentant des occurences a et des mots présentant des occurences b. Il est alors clair que ba est facteur de certains mots de $H^{@}$, ce qui contredit l'hypothèse $L' \cap L'' = \emptyset$ □

lemme 3 : $A = \bigcup_{n > 0} (ca^n b : ca^n b)$ est algébrique .

preuve du lemme

Soit B le langage algébrique : $\bigcup_{n > 0} \{ca^n bca^n b\}$,

alors $A = B \cup C$ où C est le langage reconnu par la grammaire suivante munie de l'axiome S_0 :

$S_0 \to c\ S_1\ b$

$S_1 \to a\ S_1\ a + c\ S_2 + S'_1\ b$

$S_2 \to a\ S_2\ a + b + S'_2\ b$

$S'_1 \to a\ S'_1\ a + c\ S'_2$

$S'_2 \to a\ S'_2\ a + aa$ ⊠

A ce stade on pourrait envisager d'autres classes de langages infinitaires engendrées à partir des précédentes par certains ensembles spécifiques d'opérateurs. Nous ne poursuivrons pas dans cette voie, ne disposant d'aucun critère général pour retenir certains opérateurs plutôt que d'autres. Aussi, choisissons-nous de considérer directement dans la section suivante la plus complexe des classes de langages infinitaires susceptibles d'être associés à des mécanismes de production (§3), à savoir la classe $\sum_1^1$.

II.4 - LANGAGES INFINITAIRES DE LA CLASSE $\sum_1^1$

Identifions l'alphabet X utilisé jusque à présent et l'ensemble $\mathbb{N}$ des entiers naturels. Un langage (purement) infinitaire n'est alors rien d'autre qu'un sous-ensemble de $\mathcal{F}$, l'ensemble des fonctions totales de $\mathbb{N}$ dans $\mathbb{N}$, dont l'élément f est le mot f(0)f(1).. . Le propos de la présente section est d'étudier la séparabilité de la classe des sous-ensembles $\sum_1^1$ de $\mathcal{F}$. Nous procédons au préalable, en nous inspirant de l'ouvrage de H. Rogers [12], au rappel des notions indispensables. (le lecteur familier avec les notations de cet ouvrage peut aborder directement la démonstration du théorème de non séparabilité) .

Notations et rappels

Dans la suite de ce chapitre, τ désigne une bijection récursive entre $\mathbb{N} \times \mathbb{N}$ et $\mathbb{N}$. τ^k est l'extension k - aire de τ définie par $\tau^1 = \lambda x[x]$, $\tau^2 = \tau$, et pour $k > 1$:

$$\tau^{k+1}(x_1 \ldots x_{k+1}) = \tau(\tau^k(x_1 \ldots x_k), x_{k+1}).$$

Le graphe $\tau(f)$ d'une fonction $f \in \mathcal{F}$ est l'ensemble $\{\tau(x,f(x)) \mid x \in \mathbb{N}\}$. On abrège $\tau^k(x_1 \ldots x_k)$ en $\langle x_1, \ldots, x_k \rangle$, et on note D_x l'ensemble

$$\{x_1, \ldots, x_k\} \text{ tel que } x = 2^{x_1} + \ldots + 2^{x_k} .$$

On note W_x le domaine de la fonction partielle ψ_x (à une variable entière) définie par la machine de Turing d'index x. Il existe alors une fonction récursive (totale) ρ garantissant la monovaluation de chacune des relations ψ_z^X ($\subseteq \mathbb{N} \times \mathbb{N}$) définies de la façon suivante pour $z \in \mathbb{N}$ et $X \subseteq \mathbb{N}$ ($\tilde{X}$ est la complémentaire de X dans $\mathbb{N}$) :

$$\psi_z^X = \{(x,y) \mid (\exists\, i,j)\ [\langle x,y,i,j\rangle \in W_{\rho(z)} \ \&\ D_i \subseteq X \ \&\ D_j \subseteq \tilde{X}\ ]\}\ .$$

On étend les relations précédentes en posant

$$\psi_z^{X_1,\ldots,X_n} = \psi_z^{X_1 \text{ join } (X_2 \text{ join} \ldots X_n)}$$

où $\quad X \text{ join } Y = \{j \mid j = 2i \ \&\ i \in X \text{ ou } j = 2i+1 \ \&\ i \in Y\}\ .$

Si $X_i = \tau(f_i)$, on pose $\psi_z^{f_1,\ldots,f_n} = \psi_z^{X_1,\ldots,X_n}$ (la fonction partielle à une variable entière $\psi_z^{f_1,\ldots,f_n}$ est calculée par une machine de Turing qui utilise les oracles $f_1,\ldots,f_n$).

Une fonction partielle récursive à k variables fonctionnelles et l variables entières (k, l > 0) est une fonction Φ pour laquelle il existe un entier z, dit index de Φ, tel que

$$\Phi = \lambda f_1 \ldots f_k\, x_1 \ldots x_l\ [\psi_z^{f_1 \ldots f_k}(x_1 \ldots x_l)]\ .$$

Une fonction partielle récursive à k variables fonctionnelles (k>0) est une fonction Φ' pour laquelle il existe z tel que

$$\Phi' = \lambda f_1 \ldots f_k\ [\psi_z^{f_1 \ldots f_k}(0)]\ .$$

Φ (resp. Φ') est récursive si son domaine est $\mathcal{F}^k \times \mathbb{N}^l$ (resp. $\mathcal{F}^k$).

Cette notion est étendue aux relations $R \subseteq \mathcal{F}^k \times \mathbb{N}^l$ (resp. $\mathcal{F}^k$) par l'intermédiaire de leurs fonctions caractéristiques.

$\sum_1^1$ est par définition la classe des sous-ensembles de $\mathcal{F}$ exprimables sous la forme $\{f \mid (\exists g)\ (\forall x)\ R(f,g,x)\}$ où R est une relation récursive à deux variables fonctionnelles et une variable entière. Ces ensembles ne sont qu'un cas particulier d'ensembles analytiques (de fonctions). Est un ensemble analytique tout ensemble obtenu à partir d'une relation récursive par une suite finie d'opérations de projection ou de complémentation. Les ensembles analytiques sont exactement ceux qu'expriment les formes prédicatives dans lequelles des chaînes finies de quantificateurs du premier ou du second ordre sont appliquées à des relations récursives. Les classes de complexité analytique, dont $\sum_1^1$, sont déterminées par les alternations de quantificateurs du second ordre dans les préfixes de ces expressions. Ces classes sont préservées par les transformations suivantes sur les suites de quantificateurs ($\exists^0$ et $\forall^0$ pour le premier ordre, $\exists^1$ et $\forall^1$ pour le second ordre) :

$\ldots \forall^0\forall^0 \ldots \rightarrow \ldots \forall^0 \ldots$

$\ldots \exists^0\exists^0 \ldots \rightarrow \ldots \exists^0 \ldots$

$\ldots \forall^1\forall^1 \ldots \rightarrow \ldots \forall^1 \ldots$

$\ldots \exists^1\exists^1 \ldots \rightarrow \ldots \exists^1 \ldots$

$\ldots \forall^0 \ldots \rightarrow \ldots \forall^1 \ldots$

$\ldots \exists^0 \ldots \rightarrow \ldots \exists^1 \ldots$

$\ldots \forall^0\exists^1 \ldots \rightarrow \ldots \exists^1\forall^0 \ldots$

$\ldots \exists^0\forall^1 \ldots \rightarrow \ldots \forall^1\exists^0 \ldots$

$\ldots \exists^1 \rightarrow \ldots \exists^0$

$\ldots \forall^1 \rightarrow \ldots \forall^0$

La classe $\sum_1^1$ est ainsi caractérisée par la suite réduite de quantificateurs $\exists^1\forall^0$.

Une certaine forme de normalisation vaut pour les expressions prédicatives qui décrivent des ensembles $\sum_1^1$. A chacun des ensembles E de cette classe correspond en effet un entier z (appelé $\sum_1^1$ index de E) tel que :

$$E = \{f \mid (\exists g)\ (\forall k) \sim T'_{2,0}\ (z,f,g,k)\}$$

où $T'_{2,0}$ est la relation récursive définie ci-après :

$T'_{2,0}$ (z,f,g,k) ssi $\psi_z^{f,g}(0)$ converge à la $k^{\text{ième}}$ étape dans l'énumération de $W_{\rho(z)}$ [i.e. il existe y, i, j tels que ‹0,y,i,j› figure en $k^{\text{ième}}$ position dans l'énumération de $W_{\rho(z)}$, avec $D_i \subseteq \tau(f)$ join $\tau(g) \subseteq \tilde{D}_j$] .

Nous disposons à présent de tous les éléments nécessaires à l'établissement du ...

Théorème : $\sum_1^1$ n'est pas séparable .

preuve :

Soit F la famille des fonctions f qui appartiennent à l'ensemble $\sum_1^1$ indexé par f(i) pour un certain i (dépendant de f). F est non vide puisque $\mathcal{F}$ est dans $\sum_1^1$ (la relation pleine est évidemment récursive). F est décrit par la forme prédicative :

$$F = \{f \mid (\exists i)\ (\exists g)\ (\forall j) \sim T'_{2,0}(f(i), f,g,j)\}$$

dont le préfixe $\exists^0\ \exists^1\ \forall^0$ se réduit à $\exists^1\forall^0$. Comme $\sim T'_{2,0}(f(i),f,g,j)$ est une relation récursive en (i,f,g,j), F est dans $\sum_1^1$.

Soit G l'ensemble des fonctions $h \in \mathcal{F}$ telles que pour tout i, h(i) est l'index d'un ensemble $\sum_1^1$ d'intersection non vide avec F. Remarquons qu'il existe une infinité de tels index: pour tout entier k, $F \cup \{f \mid (\forall i)\ (f(i) = k)\}$ est dans $\sum_1^1$ (lemme 4), or F est non vide et

son complémentaire $\tilde{F}$ contient une infinité de fonctions constantes (lemme 5). G est décrit par la forme prédicative

$$G = \{h \mid (\forall n)\ (\exists f)\ (\exists i)\ (\exists g)\ (\forall j)\ (\exists g')\ (\forall j')$$

$$[\sim T'_{2,0}(f(i),f,g,j) \text{ et } \sim T'_{2,0}(h(n),f,g',j')]\}.$$

La suite de réductions

$$\forall^0\exists^1\exists^0\exists^1\forall^0\exists^1\forall^0 \to \forall^0\exists^1\exists^1\exists^1\forall^0\exists^1\forall^0 \to \forall^0\exists^1\forall^0\exists^1\forall^0 \to \exists^1\forall^0\forall^0\exists^1\forall^0 \to$$

$$\exists^1\forall^0\exists^1\forall^0 \to \exists^1\exists^1\forall^0\forall^0 \to \exists^1\forall^0$$

montre que G est un ensemble de la classe $\sum_1^1$.

Soit H l'ensemble des fonctions strictement croissantes de G, et soit $\hat{H}$ l'ensemble des éléments non minimaux de H pour l'ordre lexicographique sur les fonctions (ou sur les mots). Clairement, H est non vide. Montrons que H et $\hat{H}$ sont dans $\sum_1^1$. Soit x un $\sum_1^1$ - index de G, alors H est décrit par la forme prédicative

$$H = \{f \mid (\exists g)\ (\forall k)\ (\forall i)\ (\forall j)$$

$[\sim T'_{2,0}(x,f,g,k) \text{ et } (i < j \Rightarrow f(i) < f(j))]\}$ dont le préfixe $\exists^1\forall^0\forall^0\forall^0$ équivaut à $\exists^1\forall^0$. Soit y en $\sum_1^1$ - index de H, alors $\hat{H}$ est décrit par la forme prédicative

$$\hat{H} = \{f \mid (\exists g)\ (\forall k)\ (\exists f')\ (\exists g')\ (\forall k')\ (\exists i)\ (\forall j)$$

$$[\sim T'_{2,0}(y,f,g,k) \text{ et } \sim T'_{2,0}(y,f',g',k') \text{ et } f'(i) < f(i) \text{ et}$$

$$(j < i \Rightarrow f'(j) = f(j))]\}.$$

La suite de réductions

$$\exists^1\forall^0\exists^1\exists^1\forall^0\exists^0\forall^0 \to \exists^1\forall^0\exists^1\forall^0\exists^1\forall^0 \to \exists^1\exists^1\forall^0\forall^0\exists^1\forall^0 \to \exists^1\forall^0\exists^1\forall^0 \to$$

$$\exists^1\exists^1\forall^0\forall^0 \to \exists^1\forall^0$$

montre que $\hat{H}$ est dans $\sum_1^1$.

On définit à présent $\check{H} = H - \hat{H}$. Puisque H est non vide, $\check{H} = \{\check{h}\}$ où $\check{h}$ est la fonction qui énumère dans l'ordre strictement croissant tous les index des ensembles $\sum_1^1$ d'intersection non vide avec F. Afin d'établir le théorème, il nous suffit de montrer que $\check{H}$ n'est pas dans $\sum_1^1$.

En effet, si $\sum_1^1$ est séparable, il doit exister un ensemble E de cette classe tel que $E \cap \hat{H} = \emptyset$ et $E \cap H \neq \emptyset$, d'où nécessairement $E \cap H = \{\check{h}\} = \check{H}$. Or l'intersection de deux ensembles $\sum_1^1$ est un ensemble $\sum_1^1$ (lemme 4).

Supposons donc que $\check{H}$ soit dans $\sum_1^1$ et possède l'index z. Considérons la fonction h qui à i fait correspondre le $(i+1)^{\text{ème}}$ entier dans le complémentaire de $\check{h}(\mathbb{N})$. La fonction h est bien une fonction totale : par le lemme 5, $\tilde{F}$ contient une infinité de fonctions constantes, et les $\sum_1^1$ - index des singletons constitués par ces fonctions ne sont pas énumérés par $\check{h}$. Pour tout i, h(i) est la somme de i et du plus petit entier j tel que $\check{h}(j)$ soit supérieur ou égal à i+j+1.

Par conséquent, le singleton {h} peut être décrit par la forme prédicative

$$\{f \mid (\exists f')\ (\exists g)\ (\forall k)\ (\forall i)\ (\exists j)\ (\forall l)$$

$$[\sim T'_{2,0}\ (z,f',g,k) \text{ et } (f(i) = i+j) \text{ et}$$

$$(i+j+1 \leqslant f'(j)) \text{ et } (l < j \Rightarrow f'(l) < i+l+1)]\} .$$

Etant donné la chaîne de réductions $\exists^1\exists^1\forall^0\forall^0\exists^0\forall^0 \to \exists^1\forall^0\exists^1\forall^0 \to \exists^1\exists^1\forall^0\forall^0 \to \exists^1\forall^0$, le singleton {h} est dans $\sum_1^1$. Soit x son index. Il ne nous reste plus qu'à exhiber la contradiction.

. Si $h \in F$, alors il existe un entier i pour lequel h(i) est l'index y d'un ensemble $\sum_1^1$ contenant h.

Puisque cet ensemble a une intersection non vide avec F, y appartient à $\overset{\vee}{h}(\mathbb{N})$ et ne peut pas appartenir à son complémentaire h(IN) - contradiction -

. Si $h \notin F$, alors {h} a une intersection vide avec F, donc x appartient au complémentaire de $\overset{\vee}{h}(\mathbb{N})$. Par conséquent, $x \in h(\mathbb{N})$ et il existe un entier i pour lequel h appartient à l'ensemble $\sum_1^1$ d'index h(i), d'où $h \in F$ (contradiction). □

lemme 4 : la classe $\sum_1^1$ est fermée pour l'union et pour l'intersection finies.

preuve du lemme

Soient $\overset{\cup}{\cap} \in \{\cup, \cap\}$ et $\overset{\vee}{\&} \in \{\vee, \&\}$ liés par l'équivalence

$\overset{\cup}{\cap} = \cup \iff \overset{\vee}{\&} = \vee$, alors on a l'égalité

$$\{f \mid (\exists g)\ (\forall k) \sim T'_{2,0}(x,f,g,k)\} \overset{\cup}{\cap} \{f \mid (\exists g)\ (\forall k) \sim T'_{2,0}(y,f,g,k)\}$$

$$= \{f \mid (\exists g)(\forall k)(\exists g')(\forall k')\ [\sim T'_{2,0}(x,f,g,k) \overset{\vee}{\&} \sim T'_{2,0}(y,f,g',k')]\},$$

or l'union (l'intersection) de deux relations récursives en f,g,k resp. f,g',k' est une relation récursive en f,g,g',k,k' et le préfixe $\exists^1\forall^0\exists^1\forall^0$ se réduit en $\exists^1\forall^0$ □

lemme 5 : Le complémentaire de F contient une infinité de fonctions constantes.

preuve du lemme

On raisonne par l'absurde en supposant que presque toutes les fonctions constantes appartiennent à F. Ceci entraîne que presque tous les ensembles $\sum_1^1$ contiennent au moins une fonction constante. Or pout tout i le singleton $\{\underline{i}\}$ tel que $\underline{i}(0)=0$ et $\underline{i}(j) = i$ pour $j>0$ est un ensemble $\sum_1^1$ qui ne contient pas de fonction constante □

corollaire : $\sum_1^1 \cap P(\{0,1\}^\omega)$ n'est pas séparable.

Preuve :

Soit $\Phi : \sum_1^1 \to P((0\{1\}^*)^\omega)$ l'extension ensembliste de la fonction $\phi : \mathbb{N}^\omega \to (0\{1\}^*)^\omega : \phi(f) = \psi(f(0)) \cdot \psi(f(1)) \cdot \ldots \cdot \psi(f(i)) \ldots,$

où $\psi : \mathbb{N} \to (0\{1\}^*)$ code l'entier i par $\psi(i) = 01^i$.

On montre d'abord que pour tout F dans $\sum_1^1$, $\Phi(F)$ est dans $\sum_1^1$.
Soit en effet R une relation récursive telle que

$$F = \{f \mid (\exists g)\ (\forall i)\ R(f,g,i)\},$$

on a l'égalité :

$$\Phi(F) = \{f' \mid (\exists f)(\exists g)(\exists g')(\exists g'')(\forall i)(\forall j)\ [R(f,g,i)\ \&$$

$$g''(0) = f(0)\ \&\ g''(i+1) = g''(i) + f(i+1)\ \&$$

$$g'(i) = i+1 + g''(i)\ \&$$

$$f'(g'(i)) = 0\ \&\ ((g'(i)<j\ \&\ j<g'(i+1)) \Rightarrow f'(j) = 1)]\} .$$

La conclusion désirée suit de façon immédiate.

Réciproquement, pour tout F' dans $\sum_1^1 \cap P((0\{1\}^*)^\omega)$, de façon semblable , Φ^{-1} (F') est dans $\sum_1^1$.

Soient maintenant F_1 et F_2 deux ensembles non séparables de la classe $\sum_1^1$, alors $\Phi(F_1)$ et $\Phi(F_2)$ ne peuvent pas être séparés dans $\sum_1^1 \cap P(\{0,1\}^\omega)$. Supposons le contraire. Soit H un ensemble de fonctions tel que $H \cap \Phi(F_1) = \emptyset$ et $H \cap \Phi(F_2) \neq \emptyset$ - ou vice-versa -.

Posons $H' = H \cap (0\{1\}^*)^\omega$. Par le lemme 4, H' est dans $\sum_1^1$. Il s'ensuit que $\Phi^{-1}(H')$ sépare F_1 et F_2 dans $\sum_1^1$ (contradiction) ⊠

Le chapitre 3 étudie les conséquences, en matière de test des processus communicants, de la non séparabilité de la classe $\sum_1^1$.

III. TEST DE PROCESSUS COMMUNICANTS

Après une brève introduction au calcul des systèmes communicants (§3,1.), nous montrons que la classe des langages infinitaires définis par les ensembles de suites d'actions des processus de ce calcul est presque exactement $\sum_1^1$ (§3.2), puis nous précisons nos hypothèses sur les tests (§3.3) et utilisons les constructions de la section 3.2 afin de prouver la non séparabilité de C.C.S. (§3.4).

III.2 - INTRODUCTION AU CALCUL DES SYSTEMES COMMUNICANTS

Nous présentons ci-dessous une version effective du calcul des systèmes communicants (C.C.S.), vu comme un langage applicatif de définition de systèmes de transitions. Syntaxiquement, C.C.S. est une algèbre de termes, auxquels s'identifient les états des systèmes de transitions. Sémantiquement, C.C.S. est un ensemble de règles de transitions, fixant une notion de calcul.

La syntaxe fait intervenir un ensemble d'opérateurs $\sum = \bigcup \{\sum_k \mid k \geq 0\}$ qui dépend de deux paramètres : un ensemble Λ de noms d'actions, union disjointe de trois sous-ensembles Δ , $\overline{\Delta}$ et $\{\perp, \tau\}$ respectivement indexés par $2\mathbb{N}+2$, $2\mathbb{N}+3$ et $\{0,1\}$, et un ensemble REN de renommages sur ces actions, indexé par $\mathbb{N}$.
τ symbolise un évènement invisible, produit synchronisé de deux actions $\alpha \in \Delta$ (d'index $2n+2$) et $\overline{\alpha} \in \overline{\Delta}$ (d'index $2n+3$). $\perp$ est le symbole de l'indéfini. Nous utiliserons μ et λ pour désigner les éléments de $\Lambda - \{\perp\}$ et de $\Delta \cup \overline{\Delta}$, avec la convention $\overline{\overline{\lambda}} = \lambda$. Un renommage est une fonction récursive (totale) S de Λ dans Λ telle que
$((S(\mu) = S(\mu') \Rightarrow (S(\mu) = \perp \vee \mu = \mu')) \;\&$
$(S(\tau) = \tau) \;\&\; (S(\perp) = \perp) \;\&\; (S(\lambda) \neq \perp \Rightarrow S(\overline{\lambda}) = \overline{S(\lambda)}))$.
Intervient également dans la syntaxe un ensemble X - indexé par $\mathbb{N}$ - de variables utilisées pour les définitions récursives.

Un terme t de C.C.S. est soit une variable $x(\in X)$, soit une expression $op(t_1,\ldots,t_k)$ dans laquelle $op \in \sum_k$ et les t_i sont des termes, soit une définition récursive $rec_i \vec{x} \,.\, \vec{t}$ dans laquelle $\vec{x} = (x_1 \ldots x_n)$ est un vecteur de variables, $\vec{t} = (t_1 \ldots t_n)$ est un vecteur de termes, et i est inférieur ou égal à n. Un programme C.C.S. est un terme qui a toutes ses occurences de variables liées par des occurences correspondantes des combinateurs rec_i. La signature $\sum$ de l'algèbre de termes est la suivante :
$\sum_0 = \{NIL\}$, $\sum_1 = \{\mu \mid \mu \in \Lambda\} \cup \{(S) \mid S \in REN\}$,
$\sum_2 = \{+ \,,\, |\}$, $\sum_3 = \emptyset$ pour $k \geq 3$,
où + et | sont les symboles respectifs du choix non déterministe et de la composition parallèle.

Une exécution, ou calcul clos, est une suite finie ou infinie de transitions $p_i \xrightarrow{\tau} p_{i+1}$, pouvant chacune résulter de la synchronisation des mouvements $q'_i \xrightarrow{\lambda} q'_{i+1}$ et $q''_i \xrightarrow{\overline{\lambda}} q''_{i+1}$ de deux composantes parallèles q'_i et q''_i de p_i. Un calcul clos est maximal s'il est infini ou s'il ne peut être prolongé par aucune transition. Les relations $\xrightarrow{\mu}$ sont les plus petites relations binaires (sur les programmes) définies par les axiomes et règles d'inférence 1 à 8 :

1. $\vdash \mu\, p \xrightarrow{\mu} p$

2. $p_1 \xrightarrow{\mu} q \vdash p_1 + p_2 \xrightarrow{\mu} q$

3. $p_2 \xrightarrow{\mu} q \vdash p_1 + p_2 \xrightarrow{\mu} q$

4. $p_1 \xrightarrow{\mu} q \vdash p_1 \mid p_2 \xrightarrow{\mu} q \mid p_2$

5. $p_2 \xrightarrow{\mu} q \vdash p_1 \mid p_2 \xrightarrow{\mu} p_1 \mid q$

6. $p_1 \xrightarrow{\lambda} q_1 \ , \ p_2 \xrightarrow{\overline{\lambda}} q_2 \vdash p_1 \mid p_2 \xrightarrow{\tau} q_1 \mid q_2$

7. $p \xrightarrow{\mu} q \vdash p[S] \xrightarrow{S(\mu)} q[S]$ - si $S(\mu) \neq \perp$ -

8. $t_j \left[rec_i \vec{x}.\vec{t}/x_i\right]_{i=1}^{n} \xrightarrow{\mu} q \vdash rec_j \vec{x}.\vec{t} \xrightarrow{\mu} q$

(où $t_j \left[rec_i \vec{x}.\vec{t}/x_i\right]_{i=1}^{n}$ est le terme obtenu par substitution de $rec_i \vec{x}.\vec{t}$ à chacune des occurences libres de x_i dans t_j , i=1 ... n).

On peut encore imposer aux calculs clos des programmes C.C.S. diverses contraintes d'équité (9 , 2). Etant donnée une contrainte d'équité EQ, un EQ-calcul clos est un calcul clos qui satisfait cette contrainte.

III.2 RELATIONS ENTRE C.C.S. ET LA CLASSE $\sum_1^1$

Notre premier objectif est de montrer que $\sum_1^1$ borne la complexité des ensembles de calculs et de traces des programmes C.C.S.

Définition : Soit $\mathcal{E}$ un (EQ-) calcul maximal clos du programme $p_o \mid q_o$. Une décomposition parallèle de $\mathcal{E}$ est une paire

$$\left(\left(p_o \xrightarrow{\mu'_1} p_1 \xrightarrow{\mu'_2} \ldots\ p_i \xrightarrow{\mu'_{i+1}} \ldots\right), \left(q_o \xrightarrow{\mu''_1} q_1 \xrightarrow{\mu''_2} q_2 \ldots\ q_i \xrightarrow{\mu''_{i+1}} \ldots\right)\right)$$

telle que $\mathcal{E}$ soit déductible par les règles 4 à 6 de ces deux suites de mouvements, appelées (EQ-) calculs ouverts (finis ou infinis). La trace $\eta(\mu'_1)\ \eta(\mu'_2) \ldots \eta(\mu'_i) \ldots$ du premier de ces deux calculs est définie par $\eta(\lambda) = \lambda$ et $\eta(\tau) = \varepsilon$. Par convention, un calcul ouvert d'extrêmité p_n est suffixé par la suite $(p_n \xrightarrow{\perp})^{\omega}$; de même, une trace finie est suffixée par $\perp^{\omega}$.

Nous confondrons dorénavant programmes et actions avec leur codage sur $\mathbb{N}$ (en utilisant 0 pour coder le programme indéfini).

L'entier $\langle p,\langle \mu,q\rangle\rangle$ représente le mouvement $p \xrightarrow{\mu} q$ (s'il existe). La fonction f telle que $\pi_1 \circ f(i) = p_i$ et $\pi_2 \circ f(i) = \mu_i$ pour tout i représente le calcul ouvert $p_o \xrightarrow{\mu_o} p_1 \ldots \xrightarrow{\mu_i} p_{i+1} \ldots$ (π_1 et π_2 sont les fonctions récursives de décodage des paires $\langle x,y\rangle$, i.e. $\pi_1\langle x,y\rangle = x$ et $\pi_2\langle x,y\rangle = y$) .

Afin de répondre à certains besoins techniques, nous introduisons maintenant la notion d'arbres de preuve associés aux mouvements des programmes.

Définition : L'ensemble $\mathcal{A}$ des arbres de preuve est le plus petit ensemble d'entiers qui vérifie les conditions suivantes :

- pour tous μ et p, $\langle\langle 1,0\rangle, \langle \mu p,\langle \mu,p\rangle\rangle\rangle \in \mathcal{A}$,
- si $\Theta \in \mathcal{A}$ et si $\pi_2(\Theta) \vdash \sigma$ est une instance de l'une des règles 2 à 5 ou 7 ou 8 alors $\langle\langle i,\Theta\rangle,\sigma\rangle \in \mathcal{A}$, où i est le numéro de cette règle,
- si $\Theta_1, \Theta_2 \in \mathcal{A}$ et si $(\pi_2(\Theta_1), \pi_2(\Theta_2) \vdash \sigma)$ est une instance de la règle 6 alors $\langle\langle 6, \langle\Theta_1,\Theta_2\rangle\rangle,\sigma\rangle \in \mathcal{A}$.

Clairement, $\mathcal{A}$ est un ensemble récursif, car l'ensemble des instances de chacune des règles 1 à 8 est un ensemble récursif. Cette remarque nous permet d'énoncer la ...

proposition : L'ensemble des calculs ouverts d'un programme C.C.S. est dans $\sum_1^1$ ainsi que l'ensemble des traces de ces calculs.

preuve : Soient $R_\tau(f,g,i) \subset \mathcal{F}^2 \times \mathbb{N}$ et $R_{\not\in}(f,i,j,k) \subset \mathcal{F} \times \mathbb{N}^3$ les relations définies par

$R_\tau(f,g,i) \iff$

$[g(i) \in \mathcal{A} \;\&\; \pi_2 \circ g(i) = \langle \pi_1 \circ f(i), \langle \pi_2 \circ f(i), \pi_1 \circ f(i+1)\rangle\rangle]$,

$R_{\not\in}(f,i,j,k) \iff$

$[\pi_1 \circ f(i) = \pi_1 \circ f(i+1) \;\&\; \langle j,\langle \pi_1 \circ f(i), \langle \tau,k\rangle\rangle\rangle \notin \mathcal{A}]$.

Ces deux relations sont récursives.
Soit $R(f,g) \subset \mathcal{F}^2$ (lire "f est un calcul maximal clos prouvé par g") la relation définie par
$(\forall i)(\forall j)(\forall k)$
$[(\pi_2 \circ f(i) = \tau \ \& \ R_\tau(f,g,i)) \vee (\pi_2 \circ f(i) = \bot \ \& \ R_\chi(f,i,j,k))]$.
Si $\langle f,g\rangle$ est la fonction telle que $\langle f,g\rangle(i) = \langle f(i),g(i)\rangle$ pour tout i, alors l'ensemble des paires $\langle f,g\rangle$ correspondant à des éléments (f,g) de R est dans $\sum^1_1$. Cette propriété peut être généralisée à l'ensemble des quadruplets $\langle f',f'',f,g\rangle$ tels que (f',f") soit une décomposition parallèle du calcul maximal clos f prouvé par g, et reste valide lorsque l'on restreint R(f,g) par la contrainte $\pi_1 \circ f(o) = p$, où p est un programme particulier. La proposition suit alors sans difficulté □

La prise en compte d'une contrainte d'équité n'entraîne pas de modification des propriétés précédentes. Une telle contrainte peut être décomposée sur les divers agents qui apparaissent dans un calcul clos et qui y sont identifiés par un codage des redex associés à leurs mouvements (8). (Il existe une fonction récursive qui fait correspondre à chaque transition donnée par son arbre de preuve le ou les redex des agents engagés dans cette transition). En présence d'une contrainte d'équité EQ, la relation R(f,g) de la preuve donnée ci-dessus devient
$(\forall i)(\forall j)(\forall k)(\forall l)$
$[- \ \& \ [EQ_1(f,g,i,l) \Rightarrow EQ_2(f,g,i.l)]]$
où EQ_1 et EQ_2 spécifient des propriétés relatives à l'agent l dans le $i^{ème}$ suffixe du calcul f prouvé par g. Or ces propriétés sont à notre connaissance toujours exprimables par quantification au premier ordre de relations récursives.

Afin d'étayer notre discussion ultérieure sur les tests, nous établissons dans la fin de cette section une sorte de réciproque de la proposition précédente. De nouvelles définitions sont indispensables à cet effet.

Définition : Soit $\mathcal{C}$ un calcul ouvert représenté par une fonction f. $\mathcal{C}$ est maximal si pour tous $i \in \mathbb{N}$ et $\mu \in \Lambda - \{\bot\}$, $(\pi_2 \circ f(i) = \bot)$ $\Rightarrow \sim (\pi_1 \circ f(i) \xrightarrow{\mu})$, où $p \xrightarrow{\mu}$ est l'abréviation de $(\exists q)(p \xrightarrow{\mu} q)$. $\mathcal{C}$ est un calcul simple s'il est infini $(\forall i, \pi_2 \circ f(i) \neq \bot)$ et si les implications suivantes sont vérifiées pour tous $i \in \mathbb{N}$ et $\lambda \in \Delta \cup \bar{\Delta}$:

$-(\pi_1 \circ f(i) \xrightarrow{\lambda}) \Rightarrow (\pi_2 \circ f(i) = \lambda)$

$-(\pi_2 \circ f(i) = \lambda) \Rightarrow$

$(\exists j)[(j<i) \;\&\; (\forall k)[(j < k \;\&\; k < i) \Rightarrow (\pi_2 \circ f(k) = \tau)]]$.

Un programme p est un programme simple si tout calcul ouvert maximal de ce programme est un calcul simple.

Avant d'énoncer la réciproque attendue, nous montrons pourquoi il est intéressant de l'établir en se restreignant à des programmes simples.

Définition : Soit p un programme simple et soit $\mathcal{C}$ un calcul ouvert de ce programme. Si $\mathcal{C}$ est infini, $\mathcal{C}$ est sans résidu. Si $\mathcal{C}$ est un calcul fini d'extrêmité q et si $q \xrightarrow{\lambda}$, $\mathcal{C}$ a le résidu λ. Sinon $\mathcal{C}$ a le résidu τ.

Lemme 6 : Soient p et q deux programmes simples. Si l'ensemble des traces des calculs simples de p est inclus dans l'ensemble des traces des calculs simples de q, alors à tout calcul ouvert de p correspond un calcul ouvert de q ayant même trace et même résidu.

preuve :découle directement des définitions □

Définition : Etant donnés un contexte C et un calcul maximal clos $\mathcal{C}$ de $C[\tau p]$ où p est un programme, on note $\mathcal{C} \setminus C$ la fonction qui à i fait correspondre la suite $\mathcal{C}_i$ des mouvements du $i+1^{\text{ème}}$ agent p apparu en position active dans $\mathcal{C}$ (sans résulter du déploiement d'un autre agent p) ; on pose par convention $(\mathcal{C} \setminus C)(i) = 0 \xrightarrow{\perp} 0 \xrightarrow{\perp} \ldots$ pour i supérieur ou égal au nombre total de ces agents.

Lemme 7 : Etant donnés un contexte C et un programme simple p, soit $\mathcal{C}$ un calcul maximal clos de $C[\tau p]$. Pour tout programme simple q et pour toute famille de calculs ouverts $\mathcal{C}_i$ de q tels que $\mathcal{C}_i$ et $(\mathcal{C} \setminus C)(i)$ aient même trace et même résidu, on peut former un calcul maximal clos de $C[\tau q]$ en remplaçant les suites de mouvements $(\mathcal{C} \setminus C)(i)$ de p par les suites de mouvements $\mathcal{C}_i$ de q sans autrement altérer le calcul $\mathcal{C}$.

preuve : laissée au lecteur □

L'intérêt de la notion de programme simple réside dans le fait que le lemme 7 reste valide sous toutes les contraintes usuelles d'équité. Les lemmes 6 et 7 seront utilisés pour prouver la non-séparabilité de C.C.S. en s'appuyant sur la...

proposition : Pour tout langage L de la classe $\sum_1^1$ inclus dans $\{0,1\}^\omega$, il existe un programme simple p_L tel que l'ensemble des traces de ses calculs simples soit égal à $L \cup \{0,1\}^* \perp^\omega$.

Nous donnons une preuve informelle de cette proposition en ébauchant la construction du programme p_L. Les principales étapes de la construction sont décrites par une série de lemmes démontrés en annexe.

Lemme 8 : On sait simuler en C.C.S. une pile de booléens avec test des passages à vide.

Lemme 9 : Il existe un programme simple FIN tel que l'ensemble des traces de ses calculs simples soit égal à $\{0,1\}^* \perp^\omega$.

Lemme 10 : On sait en C.C.S. simuler une machine de Turing en rendant compte de l'arrêt de la machine par l'arrêt du simulateur.

Lemme 11 : Pour tout langage L de la classe $\sum_1^1$ inclus dans $\{0,1\}^\omega$, il existe un programme simple q_L tel que l'ensemble Q des traces de ses calculs simples satisfasse $L \subset Q$ et $Q \subset L \cup \{0,1\}^* \perp^\omega$.

preuve de la proposition : $p_L = \tau\, FIN + \tau\, q_L$ ☒

III.3 HYPOTHESES SUR LES TESTS

Etant donnés deux programmes p et q, nous appelons q - expérience sur p tout calcul ouvert $\mathcal{E}_q$ intervenant dans la décomposition parallèle $(\mathcal{E}_p, \mathcal{E}_q)$ de l'un des calculs clos maximaux de p|q. Puisque les calculs ouverts sont représentés par des fonctions $f \in \mathcal{F}$, tout prédicat total $\mathcal{C}(f)$ à une variable fonctionnelle définit une bipartition {Succès,Echecs} de l'ensemble des calculs ouverts. Nous appelons critère de test un prédicat $\mathcal{C}(f)$ qui s'exprime par quantification des variables (du premier ordre) d'une relation récursive $R(f, i_1, \ldots, i_n)$. Nous supposons dans la suite avoir fixé un tel critère. Par définition,

p (ne satisfait) jamais q si l'ensemble des q -expériences $\mathcal{E}_q$ sur p est inclus dans Echecs, et p (satisfait) toujours q si l'ensemble des q - expériences $\mathcal{E}_q$ sur p est inclus dans Succès. Deux programmes p et p' sont indiscernables par test si pour tout contexte C, pour tout programme q et pour tout degré de satisfaction sat $\in$ {jamais, toujours} , on a l'équivalence C[p] sat q $\Leftrightarrow$ C[p'] sat q.

III.4 NON SEPARABILITE DE C.C.S.

Comme annoncé, nous allons appliquer ici les résultats du chapitre 2 à la théorie des tests.

théorème : Certains programmes qui diffèrent par l'ensemble des traces associées à leurs calculs ouverts (i.e. qui diffèrent par leurs suites finies ou infinies d'actions visibles) sont néanmoins indiscernables par test.

La démonstration de ce résultat, indépendant du critère de test, repose sur une série de lemmes qui utilisent les notations proposées ci-dessous.

Définition : Soit $f : \mathbb{N} \to \mathcal{F} : i \to f_i$ une suite de fonctions. On note f^{Δ} la fonction de $\mathbb{N}$ dans $\mathbb{N}$ que représente le mot $f_0(0)f_0(1)f_1(0)f_0(2)f_1(1)f_2(0)\ldots$. Inversement, si $f^{\Delta} = g$, on pose $g^{\nabla_i} = f_i$ pour tout i. On étend ces notations aux sous-ensembles F de $\mathcal{F}$: F^{Δ} est l'ensemble des éléments f^{Δ} pour lesquels f énumère une partie de F. et F^{∇_i} est l'ensemble des éléments f^{∇_i} pour lesquels f appartient à F.

lemme 12 : Etant donnés deux programmes p et q, soit $\mathcal{S}$ l'ensemble des calculs ouverts $\mathcal{E}_p$ de p tels que $\mathcal{E}_q$ appartienne à Succès (resp. Echecs) pour $(\mathcal{E}_p, \mathcal{E}_q)$ l'une des décompositions parallèles de calculs clos maximaux de p|q, alors $\mathcal{S}$ est dans Σ^1_1.

preuve : Soit DEC (f,f',f",g) la relation récursive qui est vérifiée si et seulement si le calcul maximal clos f prouvé par g admet la décomposition parallèle (f',f") - voir la preuve de la première proposition donnée dans la section 3.2 - On a alors
$\mathcal{S} = \{f' \mid (\exists f)(\exists f")(\exists g)\ [\mathcal{C}(f") \ \&\ DEC(f,f',f",g)]\}$,

où $\mathcal{C}(f")$ - resp. $\sim \mathcal{C}(f")$ - exprime le succès de f" par un prédicat de la forme

$(\forall\exists_1\ i_1)\ldots(\forall\exists_n\ i_n)\ R(f", i_1, \ldots, i_n)$.

Or chacun des préfixes $(\exists f)(\exists f")(\exists g)\ (\forall\exists_1\ i_1)\ \ldots\ (\forall\exists_n\ i_n)$,

où $\forall\exists_j \in \{\forall, \exists\}$, admet la forme réduite $\exists^1 \forall^0$ □

Lemme 13 : Pour tout contexte C et pour tout programme p, si Q est un ensemble $\sum^1_1$ de calculs ouverts de C[τp], alors $\{(\mathcal{C} \setminus C)^\Delta \mid \mathcal{C} \in Q\}$ est dans $\sum^1_1$.

preuve : On décore les règles opérationnelles de C.C.S. par un jeu d'exposants et d'indices, respectivement utilisés pour totaliser le nombre des agents p apparus à une certaine étape d'un calcul et pour distinguer ces agents les uns des autres. Les indices, propres aux agents p, sont hérités par leurs descendants (les autres termes ne portent pas d'indice). On introduit les variantes suivantes des règles d'action :

$(\tau\, p)^{\langle i\rangle} \xrightarrow{\tau} (p)^{\langle i+1\rangle}_{\langle i+1\rangle}$,

$(\alpha\, q)^{\langle i\rangle}_{\langle j\rangle} \xrightarrow{\alpha} q^{\langle i\rangle}_{\langle j\rangle}$. Les règles d'inférence sont adaptées en conséquence : $(q|r)^{\langle i\rangle} \xrightarrow{\tau} (q'|r')^{\langle i\rangle}$ se déduit par exemple de $q^{\langle i\rangle} \xrightarrow{\alpha} q'^{\langle i\rangle}$ et $r^{\langle i\rangle} \xrightarrow{\bar{\alpha}} r'^{\langle i\rangle}$.

Soit Q' l'ensemble des suites de mouvements qui se réduisent par oubli des indices et exposants à des calculs de Q. Si Q est dans $\sum^1_1$, alors Q' l'est également, et une vérification de routine permet d'établir le lemme à partir de cette propriété □

Lemme 14 : Il existe dans la classe $\sum^1_1$ deux ensembles non séparables H et K tels que H^Δ et K^Δ soient non séparables.

preuve : Nous connaissons déjà dans la classe $\sum^1_1$ deux ensembles non séparables H et $\hat{H}$ tels que H contienne $\hat{H}$ et que $H - \hat{H}$ soit un singleton $\{\check{h}\}$. Nous prouvons que les conditions du lemme sont vérifiées pour K égal à $\hat{H}$.

Nous montrons d'abord que pour tout A dans $\sum^1_1$, A^Δ est aussi dans $\sum^1_1$. Pour cela, on note $\Delta(i,j)$ le rang occupé par la paire (i,j) dans l'énumération usuelle (o,o),(o,1),(1,o),(o,2),(1,1),(2,o)... de $\mathbb{N} \times \mathbb{N}$.

Clairement, Δ est une fonction récursive. Soit alors

$A = \{f \mid (\exists g)(\forall i)\ R_A(f,g,i)\}$ où R_A est récursive. On a

$A^\Delta = \{f' \mid (\forall j)(\exists f)(\exists g)(\forall i)\ [R_A(f,g,i)\ \&\ f'(\Delta(j,i)) = f(i)]\}$.

Le préfixe $\forall^0 \exists^1 \exists^1 \forall^0$ de cette expression se réduit comme prévu à $\exists^1 \forall^0$.

Nous adoptons pour la suite de la démonstration le procédé du raisonnement par l'absurde, en supposant l'existence dans $\sum^1_1$ d'un ensemble E tel que $E \cap H^\Delta \neq \emptyset$ et $E \cap K^\Delta = \emptyset$. Puisque $E \cap H^\Delta$ est dans $\sum^1_1$ (lemme 4), nous pouvons supposer $E \subset H^\Delta$ sans perte de généralité. Maintenant, pour toute fonction $e \in E$, si l'on pose $e_i = e^{\nabla i}$ pour tout i, l'ensemble $\{e_o \ldots e_i \ldots\} \cap (H - K)$ est nécessairement le singleton $\{\overset{\vee}{h}\}$, c.a.d. l'ensemble H - K ! L'idée consiste alors à utiliser l'identité

$$H - K = \{h \mid (\exists e \in E)(\forall i)\ [(\sim(e_i \in K)) \Rightarrow (e_i = h)]\}$$

pour montrer que ce singleton est dans $\sum^1_1$, d'où contradiction puisque H et K sont non séparables.

Posons $E = \{e \mid (\exists g)(\forall j)\ R_E(e,g,j)\}$ et $K = \{h \mid (\exists f)(\forall j)\ R_K(h,f,j)\}$ où R_E et R_K sont deux relations récursives. On a $H - K =$

$$\{h \mid (\exists e)(\exists g)(\forall j)(\forall i)(\exists e_i)(\forall k)(\exists f)(\forall m)(\forall n)$$

$$[R_E(e,g,j)\ \&\ e_i(k) = e(\Delta(i,k))\ \&\ [R_K(e_i,f,m) \vee (e_i(n) = h(n))]]\}.$$

Le préfixe $\exists^1 \exists^1 \forall^0 \forall^0 \exists^1 \forall^0 \exists^1 \forall^0 \forall^0$ de cette forme prédicative se réduit en $\exists^1 \forall^0$, d'où la démonstration ∎

Corollaire : Soit $\Phi : \sum^1_1 \longrightarrow P((0\{1\}^*)^\omega)$ l'extension ensembliste de la fonction $\phi : \mathbb{N}^\omega \longrightarrow (0\{1\}^*)^\omega$:

$\phi(f) = \psi(f(0)) \bullet \psi(f(1)) \bullet \ldots \bullet \psi(f(i)) \ldots$, où

$\psi : \mathbb{N} \to (0\{1\}^*)$ code l'entier i par $\psi(i) = 0\ 1^i$.

Alors $(\Phi(H))^\Delta$ et $(\Phi(K))^\Delta$ sont deux ensembles non séparables de la classe $\sum^1_1$.

(même démonstration que ci-dessus, en utilisant le fait que $\Phi(H)$ et $\Phi(K)$ sont deux ensembles non séparables de la classe $\sum_1^1$)

Forts de cette série de lemmes, nous abordons maintenant la partie centrale de la ...

Démonstration du théorème : Soient $P = \Phi(H)$ et $Q = \Phi(K)$, ou H et K sont les ensembles de fonctions du lemme 14. Puisque P et Q sont dans $\sum_1^1$, on peut faire correspondre à ces ensembles de chaines binaires deux programmes simples p et q admettant respectivement $P \cup \{0,1\}^* \perp^\omega$ et $Q \cup \{0,1\}^* \perp^\omega$ comme ensembles des traces de leurs calculs simples. Notre intention est de montrer que τ_p et τ_q ne sont alors pas distinguables par test, ce qui suffit à prouver le théorème puisque P - Q est non vide.

Supposons τ_p et τ_q séparables. Soient alors C,t et sat les contexte, programme de test et degré de satisfaction qui les distinguent, i.e. vérifient

$C[\tau_p]$ sat t $\langle\not\Rightarrow\rangle$ $C[\tau_q]$ sat t.

En vertu des lemmes 6 et 7, puisque Q est inclus dans P, on a

$C[\tau_p]$ sat t $\Rightarrow$ $C[\tau_q]$ sat t.

Donc $C[\tau_q]$ sat t $\not\Rightarrow$ $C[\tau_p]$ sat t.

Soit $\mathcal{P}$ l'ensemble des calculs ouverts $\mathcal{C}_p$ de $C[\tau_p]$ pour lesquels l'un des calculs clos maximaux de $(C[\tau_p] \mid t)$ se décompose en $(\mathcal{C}_p, \mathcal{C}_t)$ avec $\mathcal{C}_t$ dans Echecs si sat = toujours, ou dans Succes si sat = jamais. Par les lemmes 12 et 13, l'ensemble $\mathcal{P}'$ égal à $\{(\mathcal{C} \setminus C)^\Delta \mid \mathcal{C} \in \mathcal{P}\}$ est dans $\sum_1^1$. Pour toute fonction f, soit traces(f) la fonction telle que $(\text{traces}(f))^{\nabla i}$ soit égale pour tout i à la trace du calcul ouvert représenté par $f^{\nabla i}$ (s'il existe). Alors clairement, l'ensemble $\mathcal{P}'' = \{\text{traces}(f) \mid f \in \mathcal{P}'\}$ est dans $\sum_1^1$.

Maintenant, puisque p et q sont des programmes simples qui ont les mêmes traces finies, l'assertion

($C[\tau_q]$ sat t $\not\Rightarrow$ $C[\tau_p]$ sat t) entraine par le lemme 7 la conclusion $\mathcal{P}'' \cap (Q \cup \{0,1\}^* \perp^\omega)^\Delta = \emptyset$, tandis que $\mathcal{P}'' \cap (P \cup \{0,1\}^* \perp^\omega)^\Delta \neq \emptyset$ par construction. Soit R l'ensemble de traces infinies donné par

$\{g \mid (\exists f \in \mathcal{S}^{\omega})(\exists i)(\forall j)(g = f^{\nabla_i} \ \& \ g(j) \neq \perp)\}$.

R et R^{Δ} sont deux ensembles de la classe $\sum^1_1$, et R^{Δ} sépare nécessairement P^{Δ} de Q^{Δ} , ce qui est impossible ⊠

Remarque : La non séparabilité a été démontrée pour la version standard (non équitable) de C.C.S. . Un examen détaillé montre que la démonstration qui a été donnée reste valide en remplaçant la notion de calcul clos par celle d'EQ-calcul clos pour l'une quelconque des contraintes usuelles d'équité.

IV. QUELQUES CONCLUSIONS

Parvenus au terme d'une étude dans laquelle nous avons amalgamé langages et processus, il nous reste à justifier ce rapprochement et à tirer les enseignements de nos résultats quant aux modèles du parallèlisme.

L'exemple de la séparabilité montre que certaines propriétés essentielles des classes de langages infinitaires sont à ce jour insuffisamment explorées : celles dont la transposition finitaire est triviale. La considération de ces propriétés peut être motivée par des questions liées aux théories des processus communicants et de leur observation, sources possibles d'un enrichissement de la problèmatique des langages. (Le peu de méthodes disponibles pour former des classes de langages infinitaires à partir de classes finitaires montre en outre l'intérêt des calculs de processus communicants pour la définition directe de ces classes).

Le rapport entre (classes de) processus et (classes de) langages infinitaires est plus étroit que ne l'indiquent les remarques précédentes. Nous donnons ci-dessous quelques arguments à l'appui de la thèse selon laquelle l'extension d'un processus est ou s'apparente à un langage infinitaire dans toute sémantique "observationnelle" des calculs de processus communicants.

Un temps infini est certainement nécessaire au constat du refus continu d'une action par un processus. Si les refus de ce type sont observables, alors les suites infinies d'actions d'un processus le sont aussi. car leur observation ne demande pas davantage de temps. Les domaines algébriques (4,14) ne sont donc pas appropriés à la construction de sémantiques "observationnelles" des calculs de processus. Tout objet d'un domaine algébrique est par définition la borne supérieure de ses approximants finis, représentant autant de propriétés finies. Or on connait des processus qui présentent exactement les mêmes propriétés finies (mêmes suites et ensembles d'actions possibles, garanties ou refusées...) et qui diffèrent uniquement par leurs suites infinies d'actions visibles (e.g. les processus p et q considérés dans la preuve du dernier théorème).

Comme la notion d'algébricité, la notion d'ordre d'amélioration (4,14) qui sous-tend la plupart des travaux classiques en sémantique dénotationnelle nous semble mal appropriée à la modélisation des calculs de processus. Un préordre fondé sur l'observation est défini par un critère qui permet de juger, étant donnés deux processus, si l'un d'eux coopère plus ou mieux que l'autre avec un processus tiers (l'expérimentateur). Ce critère peut être simplement l'inclusion des ensembles de comportements respectivement permis au processus tiers par les processus soumis à la comparaison : le préordre considéré est alors un préordre d'implémentation. Dans tout autre cas de figure, le critère utilisé est un critère de test car il est nécessaire d'attribuer à chaque exécution du processus expérimentateur un poids dans une échelle de succès ou d'échec.

Les processus qui ne sont pas distingués par test ont alors même représentation dans le modèle du préordre, or nous avons montré que certains de ces processus diffèrent par leurs suites infinies d'actions visibles.

Dans le cas d'un préordre d'implémentation, le domaine sémantique le plus naturel est un domaine ensembliste, ordonné par inclusion, dans lequel les éléments des ensembles représentent les propriétés observables des processus. Puisque les suites infinies d'actions visibles des processus sont des propriétés observables, ces ensembles sont ou s'apparentent à des langages infinitaires. Le modèle équitable de CCS présenté dans (2) a été construit selon ce principe.

ANNEXE

Nous donnons ici les preuves des lemmes 8 à 11. La notation $(x_1 = t_1, \ldots, x_n = t_n)$ doit être interprétée comme $rec_1(x_1 \ldots x_n) \circ (t_1 \ldots t_n)$. S_o, S_1, S_2 et S_3 désignent les renommages les moins définis tels que $S_o(\alpha) = S_1(\alpha) = S_2(\alpha) = \alpha$ et $S_3(\alpha) = \alpha'$ pour $\alpha \in \{0,1\}$, $S_0(2) = 2$ et $S_2(2) = S_3(2) = 2'$.

preuve du lemme 8 Si 2 est le signal de détection du passage à vide, le comportement recherché est celui du programme PILE donné par la définition récursive

$$(z = ((1.(x_1|x[S_3]) + 0.(x_o|x[S_3]))[S_2] \mid 2'.\overline{2}.z)[S_o],$$

$$x_o = \overline{0}.(2'.x + 1'.x_1 + 0'.x_o) + 0.(\overline{0}'.x_o) + 1.(\overline{0}'.x_1),$$

$$x_1 = \overline{1}.(2'.x + 1'.x_1 + 0'.x_o) + 0.(\overline{1}'.x_o) + 1.(\overline{1}'.x_1),$$

$$x = \overline{2}.NIL + 0.(x_o|x[S_3]) + 1.(x_1|x[S_3])).$$

preuve du lemme 9 Le comportement recherché est celui du programme donné par la définition récursive

$$(z = (PILE[S_3] \mid x_o)[S_1],$$

$$x_o = \tau.\overline{0}'.x_1 + \tau.\overline{1}'.x_1 + \tau.y_1,$$

$$x_1 = \tau.\overline{0}'.x_1 + \tau.\overline{1}'.x_1 + \tau.y_o,$$

$$y_o = 0'.0.y_o + 1'.1.y_o + 2'.y_1,$$

$$y_1 = \tau.y_1).$$

preuve du lemme 10 Il suffit de construire un programme C.C.S. dans lequel coopèrent un automate d'états finis, qui représente le programme de la machine et mémorise son état programme, et deux piles qui représentent la bande de la machine, scindée en ses portions à gauche et à droite de la tête de lecture (PILE$[T_g]$ et PILE$[T_d]$). Or la correspondance entre automates d'états finis et systèmes de définitions récursives est évidente. L'arrêt de la machine avec le résultat n peut en particulier

être simulé par une transition C.C.S. dont le terme d'arrivée est équivalent par bisimulation à un terme de la forme $(PILE[T_g] \mid (z=start \bullet ...,...) \mid (PILE_n[T_d])$, dans lequel $(z=start \bullet ...,...)$ est l'automate d'états finis en attente du lancement d'une nouvelle exécution par la commande start, et $PILE \xrightarrow{\overline{0}} (\xrightarrow{\overline{1}})^{n+1} PILE_n$.

preuve du lemme 11 A chaque langage A de la classe Σ^1_1, on sait faire correspondre une relation récursive R_A telle que
$A = \{f \mid (\exists h)(\forall i)\ R_A(f[0...i], h[0...i])\}$, où $g[0...i] = \langle g(o), \langle g(1), ... g(i)\rangle ...\rangle$.

Notons $\hat{h}(j)$ la chaîne $1^{1+h(j)}$ et posons :
h jusques i = $\hat{h}(0).0.\hat{h}(1). \ ... \ . \hat{h}(i).0$.
Si f varie dans $\mathbb{N} \to \{0,1\}$, l'ensemble de chaînes
$\{f(0).f(1). \ ... \ .f(i).$ h jusques i $\mid R_L(f[0...i], h[0...i])\}$, où R_L correspond au langage L, est récursif. Soit ϕ_L sa fonction caractéristique, et soit PHI_L l'automate $(z = start \bullet ...,...)$ de la machine de Turing qui calcule ϕ_L.

Alors le programme q_L a la structure
$(PILES_g \mid (PHI_L \mid CONTROLE) \mid PILES_d)\ S_1$,

où $PILES_g = PILE[T_g] \mid PILE[F_g] \mid PILE[H_g]$

et $PILES_d = PILE[T_d] \mid PILE[F_d] \mid PILE[H_d]$.

Le couple $(PILE[T_g], PILE[T_d])$ simule la bande de la machine de Turing. Les autres jeux de piles appariées simulent deux bandes auxiliaires utilisées pour représenter respectivement f(0) ... f(i) et h jusques i. Afin de fixer les notations, posons $S_y(\alpha) = \alpha^s_y$ pour $S \in \{T,F,H\}$ et $y \in \{g,d\}$. Le paragraphe suivant donne quelques indications sur l'algorithme du programme CONTROLE donné à la fin du texte.

A l'initialisation, la metavariable i vaut 0 et toutes les piles sont vides. Au début de la $n + 2^{ème}$ étape, la metavariable i vaut n + 1, les données f(o)...f(n) et h jusques n sont rangées dans les piles de gauche réservées à f et h, et les autres piles sont vides. Pour chacune des valeurs successives de i, les opérations suivantes sont effectuées :

1 - choisir aléatoirement f(i) et l'empiler à gauche,

2 - choisir aléatoirement h(i) et empiler à gauche $\hat{h}(i).0$,

3 - recopier h jusques i sur la bande de la machine de Turing (PILE[T_d]) et dupliquer cette donnée dans la pile de droite réservée à h,

4 - recopier f(0) ... f(i) sur la bande de la machine de Turing et dupliquer cette donnée dans la pile de droite réservée à f,

5 - lancer l'exécution de PHI_L (commande start) et attendre le résultat ; si ce résultat est 0 (codé par 1.0 dans la pile), adopter le comportement purement divergent (τ^ω), sinon passer à la suite,

6 - transférer le contenu des piles de droite dans les piles de gauche,

7 - imiter le comportement f(i).

CONTROLE est donné par la définition récursive

$$(x_1 = \overline{0}^{\,f}_g \cdot x_2 + \overline{1}^{\,f}_g \cdot x_2 ,$$

$$x_2 = \overline{1}^{\,h}_g \cdot y_2 , \; y_2 = \overline{1}^{\,h}_g \cdot y_2 + \overline{0}^{\,h}_g \cdot x_3 ,$$

$$x_3 = 0^h_g \cdot \overline{0}^{\,h}_d \cdot \overline{0}^{\,t}_d \cdot x_3 + 1^h_g \cdot \overline{1}^{\,h}_d \cdot \overline{1}^{\,t}_d \cdot x_3 + 2^h_g \cdot x_4 ,$$

$$x_4 = 0^f_g \cdot \overline{0}^{\,f}_d \cdot \overline{0}^{\,t}_d \cdot x_4 + 1^f_g \cdot \overline{1}^{\,f}_d \cdot \overline{1}^{\,t}_d \cdot x_4 + 2^f_g \cdot x_5 ,$$

$$x_5 = \overline{\text{start}} \cdot 1^t_d \cdot (0^t_d \cdot y_5 + 1^t_d \cdot 0^t_d \cdot x_6) , \; y_5 = \tau . y_5 ,$$

$$x_6 = 0^h_d \cdot \overline{0}^{\,h}_g \cdot x_6 + 1^h_d \cdot \overline{1}^{\,h}_g \cdot x_6 + 2^h_d \cdot y_6 ,$$

$$y_6 = 0^f_d \cdot \overline{0}^{\,f}_g \cdot y_6 + 1^f_d \cdot \overline{1}^{\,f}_g \cdot y_6 + 2^f_d \cdot x_7 ,$$

$$x_7 = 0^f_g \cdot \overline{0}^{\,f}_g \cdot 0 \cdot x_1 + 1^f_g \cdot \overline{1}^{\,f}_g \cdot 1 \cdot x_1)$$

REFERENCES

1. Boasson L., Nivat M.,"Adherences of Languages"
Journal of Computer and System Sciences 20 (1980), 285-309

2. Darondeau Ph.,"About Fair Asynchrony"
- à paraitre dans T.C.S. -

3. Eilenberg S.,"Automata, Languages and Machines",
Vol. A, Academic Press (1974)

4. Guessarian I.,"Algebraic Semantics"
Springer-Verlag LNCS 99 (1981)

5. Gonzales Alvadaro C.A.,"Le Mélange et le Mélange Itératif : les Opérateurs Concurrents"
Thèse de 3ème cycle, Université Paris VII (1984)

6. Hennessy M., De Nicola R., "Testing Equivalences for Processes"
Theoretical Computer Science 34 (1984) 83-134

7. Hennessy M., "Synchronous and Asynchronous Experiments on Processes", Information and Control 59 (1983) 36-83

8. Hennessy M., "Modelling Finite Delay Operators"
University of Edinburgh, Dept of Computer Science, Internal Report CSR 153-183(1983)

9. Hennessy M., "An Algebraic Theory of Fair Asynchronous Communicating Processes"
prochainement présenté à ICALP 85

10. Milner R., "Fully Abstract Models of Typed Lambda-Calculi"
Theoretical Computer Science 4 (1977) 1-23

11. Milner R., "A Calculus of Communicating Systems"
Springer-Verlag LNCS 92 (1980)

12. Rogers H., "Theory of Recursive Functions and Effective Computability"
Mc Graw-Hill (1967)

13. Shaw A.C., "Software Descriptions with Flow Expressions"
IEEE Transactions on Software Engineering 3 (1978) 242-254

14. Stoy J.E., "Denotational Semantics"
M.I.T. Press (1977)

REMERCIEMENTS

Je tiens à remercier ici L. KOTT et S. YOCCOZ, avec lesquels a été conduite une étude préliminaire de la notion de séparabilité forte pour quelques classes de la hiérarchie.

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique